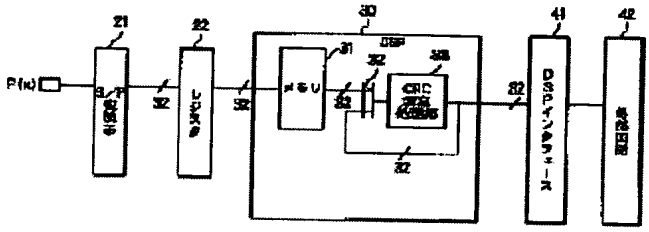
METHOD FOR DETECTING TRANSMISSION CODE ERROR

Publication number: JP9149007
Publication date: 1997-06-06
Inventor: OKUNO MIKIFUMI
Applicant: OKI ELECTRIC IND CO LTD
Classification:
- international: ***G06F11/10; H03M13/00; H04L1/00;*** **G06F11/10; H03M13/00; H04L1/00;** (IPC1-7): H04L1/00; G06F11/10; H03M13/00
- European:
Application number: JP19950304552 19951122
Priority number(s): JP19950304552 19951122

Abstract of **JP9149007**

PROBLEM TO BE SOLVED: To conduct cyclic redundancy check(CRC) operation at a high speed in a short time by using a software program of a digital signal processing(DSP). SOLUTION: A weight required for the CRC operation in a DSP 30 based on an equation of a residue R (x) is stored in advance in a memory 31 in a table form for each bit providing an arithmetic result. The DSP 30 is used, a table in the memory 31 is referenced with respect to transmission data P (x) received in parallel to detect data at an r-bit location required for the CRC operation. The DSP 30 divides the detection result of r-bit by 2<r/2> , then r/2 is substituted to the r and the division is repeated by a number of times of log2 (r)-1} to conduct EXOR by shift of 2<r> bits for r-number of times with respect to the detection result of r-bit to obtain the residue R (x).



Japanese Patent Application Laid-Open No. H9-149007 (Reference 2)

(54) [Title of the Invention]

A METHOD OF DETECTING TRANSMISSION CODE ERRORS

(57) [Abstract]

[Problem to be solved]

To perform a CRC (cyclic redundancy check) operation at a high speed in a short time by using a software program of a DSP (digital signal processor).

[Solution]

A weight required for the CRC operation performing in a DSP 30 based on an equation of a remainder R (x) is stored in advance in a memory 31 in a table form for each bit of an operation result. The r-bit positional data required for the CRC operation for the transmission data P(x) input in parallel is detected using the DSP 30 with reference to the table in the memory 31. The DSP 30 divides the r-bit detection data by $2^{r/2}$ and repeats the division $(\log_2(r)-1)$ times with r taken to be r/2 to perform an EXOR operation between the r-bit detection result and 2^r bit shift r times to obtain the remainder R(x).

[Detailed Description of the Invention]

[0001]

[Field of the Invention]

The present invention relates to a method of detecting transmission code errors in the subscriber line transmission by a cyclic redundancy check (hereinafter referred to as "CRC") operation by using a digital signal processing device (hereinafter referred to as "DSP").

[0002]

[Conventional Art]

Figure 2 is a drawing describing the CRC which is one of methods of detecting transmission code errors, for example, in the subscriber line transmission. The CRC is one of the methods in which redundant bits are added to a data to detect errors during the transmission of information, and used in the communication protocol in the subscriber line transmission. In the CRC, a transmission data P(x) expressed by a multinomial is multiplied by x^r (where, r is a positive integer) and the $x^r \cdot P(x)$ is divided by a generator polynomial to obtain a remainder R(x) of r-bit expressed by a multinomial. An information transmission side adds the obtained remainder R(x) to the $x^r \cdot P(x)$ as a check bit and sends it out as transmission information T(x). An information reception side divides the received information R(x) by the generator polynomial G(x) to obtain a remainder. If the remainder is zero, it is regarded that the information is correctly transmitted. If there is a remainder, it is regarded that the transmitted information has an error. A CRC processing which detects such an error has hitherto realized by a hardware because it can be relatively easily realized.

[0003]

[0004]

Figure 4 shows an example of configuration of the CRC-12 operation circuit in Figure 3. The CRC-12 operation circuit includes a dividing circuit composed of 12-bit registers 11-1 to 11-12 and EXOR gates 2, 12-1, 12-2, 12-3, ..., and latch circuits 13-1 to 13-12 holding arithmetic results of each bit connected to the output of the dividing circuit. In the CRC processor in Figure 3, the value of the remainder R(x) output as a result of the division of the input transmission data P(x) by the CRC operation circuit 3 is expressed by an EXOR between bit (bit of "1") with a weight represented by the generator polynomial G(x) and an LSB bit in serial CRC operation data. In Figure 4, for the subscriber line transmission, a series of transmission data P(x) is sent in the form of a multiframe. At the time of starting the multiframe, the registers 11-1 to 11-12 are cleared and then the transmission data P(x) is input from the LSB side bit by bit, and all of 12 bits are input by the shift operation of the registers 11-1 to 11-12. The value of the operation result is obtained by performing an EXOR operation between one bit of input transmission data P(x) and LSB of the CRC operation data for all bits (12 times) at a position corresponding to bit with a weight expressed by the generator polynomial G(x). The latch circuits 13-1 to 13-12 are in an enable state at the time of finish of the multiframe and hold each bit of CRC operations as operation results.

[0005]

When the following multiframe transmission data P(x) is input, the CRC-12 operation circuit performs CRC operations in the same manner as the above. The LSB of the operation result is fed back to input side via EXOR gate 2, which means that continuously input transmission

[0007]

In the CRC-32 operation in Figure 5, when processing is started, the transmission data P(x) is serially input. At step S1, a determination is made as to whether the data of 32 bits have been input. When the data of 32 bits have been input, the process proceeds to step S2. At step S2, a 32-bit input data "crddat" is stored in a memory and called each time a CRC operation result CRC[i] is calculated (32 times in total). At step S2, first, i = 0 in the 32-bit input data "crddat" stored in the memory is read and sent to step S3. At step S3, a mask pattern CRC[i] representing a bit position where "1" is set up in the generator polynomial G(x) is generated and sent to step S4. At step S4, the logical AND (hereinafter referred to as "AND") between the input data "crddat" and mask pattern CRC[i] is carried out and the operation result "result[i]" is sent to step S5. At step S5, an LSB of one bit in the CRC operation data and a value stored in the register (or, data of each bit position based on the generator polynomial G(x)) are EXORed and the operation result "result 1" is sent to step S6. At step S6, the operation results "result 1" of all of 32 bits are shifted by one bit to the right, and then an LSB of one bit is fed back to the input register at step S7 and the process advances to step S8. At step S8, after one (1) has been added to i, the process proceeds to step S9. A determination is made as to whether CRC processing is conducted 32 times. If a 32-time processing has not been finished, the process returns to step S3 to repeat the above processing. If the 32-time processing has been finished, the CRC-32 processing is terminated.

operation in a digital signal processor based on the equation of the remainder R(x) derived from the generator polynomial G(x) is stored in advance in a memory in the digital signal processor in the table form for each bit of the operation results. The r-bit position data required for the CRC operation is detected with respect to the transmission data P(x) using the digital signal processor with reference to the table, the r-bit detection result is divided by $2^{r/2}$ and the division is repeated $(\log_2(r)-1)$ times with r taken to be r/2 to perform an EXOR operation between the r-bit detection result and 2^r bit shift r times to obtain the remainder R(x). According to the present invention, the remainder R(x) is obtained in the above manner, so that position data required for the CRC operation with respect to the parallel transmission data P(x)is detected using the table in the memory. The detection result and 2^r-bit shift are EXORed i times to calculate the remainder R(x).

[0010]

[Embodiments of the Invention]

Figure 1 shows an example of configuration of a CRC processor used in a method of detecting transmission code errors by CRC operation according to the embodiment of the present invention. The CRC processor is a device for performing CRC-32 processing by DSP, for example, in a subscriber circuit and includes an S/P converter 21 which converts serially input transmission data P(x) into 32-bit parallel data. A register 22 for holding data is connected to the output of the S/P converter 21, and the output of the register 22 is connected to a DSP 30. The DSP 30 includes a memory 31 for holding the 32-bit data output from the register 22, the output of which is connected to one of the input side of an EXOR operation section 32, the output of which is

operation stored in advance in the memory 31 in Figure 1. Figures 9 and 10 show the examples of EXOR processing in the process flow in Figure 6. Referring to Figure 1, Figure 6 and Figures 7 to 10, a description is made of a method of detecting transmission code errors using the CRC processor in Figure 1 according to the present embodiment. When a serial transmission data P(x) is input into the CRC processor in Figure 1, the data is converted into 32-bit parallel data by the S/P converter 21 and then held in the register 22. The tables shown in Figures 7 and 8 need to be stored in advance in the memory 31 in the DSP 30 to perform the CRC-32 processing of the 32-bit parallel data read from the register 22 in parallel by DSP 30. While the data in the table shown in Figures 7 and 8 are shown by HEX (hexadecimal) data on the right end, the data in the process flow in Figure 6 are shown by CRC[i] (i = 0 to 31). For example, in a remainder "AMARI_31" on the left end, "1" is set up at 00 bit, 02 bit and 03 bit in 00 bit to 03 bit. This is expressed as D by HEX data. At G(x) bit to R(x) bit, "1" is set up at G(x) bit, this is expressed as one (1) by HEX data. Thus, HEX data of remainder for performing the CRC-32 processing in parallel are stored in the form of a table in advance in the memory 31 in Figure 1.

[0013]

In the process flow in Figure 6, at step S11, the DSP 30 starts processing and determines whether parallel data of 32 bits have been input from the register 22. When the data has been input, the process proceeds to step S12. At step S12, data of bit i is set to 0 and a bit shift quantity "shift" is set to 2^{16}, and the process proceeds to step S13. At step S13, the CRC processing section 33 performs an AND operation between 32-bit tape data "CRC[i]" read from the memory

taken to be a third operation result "result". The data shifted by two bits to the right and the third operation result "result" are EXORed to be taken to be a fourth operation result "result". Furthermore, the data shifted by one bit to the right and the fourth operation result "result" are EXORed to be taken to be a fifth operation result "result". At step S17, when shift quantity "shift" is zero, the process proceeds to steps S18 and S19. When even numbers of positive logic values "1" are included in initially input 32-bit data "result 1", the fifth operation result is "0", and when odd numbers of positive logic values "1" are included, the fifth operation result is "1", which is equivalent to the all bits of the input data "result 1" being EXORed. Thus, 32-bit input data "crddat" shifted by 16, 8, 4, 2 and 1 bits to the right repetitively and the data before shifting are EXORed 32 times respectively through steps 19 and 20, thereby enabling the values of output i bit (i = 0 to 31) to be obtained in step S19. The other 29-bit-th to 00-bit-th data shown in Figures 7 and 8 are processed by the same method as described in Figures 9 and 10 to obtain the values of output i bit (i = 0 to 31) in the total flow chart. The other 30-bit-th to 00-bit-th data are processed by the same method as described in Figures 9 and 10 and EXORed 32 times in total, thereby terminating a first processing of the CRC-32.

[0016]

Since a subscriber line transmission is used in the present embodiment, the transmission data P(x) is continuously input into the CRC processor on a P(x) frame basis, so that performing the EXOR between the remainder R (x) obtained in DSP 30 by the above procedures and newly input data of 32 bits is equivalent to the CRC-32 operation of continuously input data following previous input data. For example,

Incidentally, the present invention is not limited to the above embodiments, but may be modified into various forms. The following are examples of modifications.

[0018]

(i) In the present embodiment, a description is made of the CRC-32 processing by the DSP 30 in a subscriber circuit. Even if the degree of the generator polynomial G(x) in the CRC operation is increased, the same operational effect as in the above embodiments can be expected.

(ii) The CRC processor in Figure 1 may be differently configured. In addition, the table in Figure 7 stored in the memory 31 may store data in a different form.

(iii) While the present embodiment describes a method of detecting transmission code errors in subscriber line transmission, the present invention can be adapted for a method of detecting transmission code errors except for a subscriber line.

[0019]

[Advantages of the Invention]

As described above, according to the present invention, the weight required for CRC computation is stored in a memory in the table form for each bit of the operation result, the position data of r bit required for the CRC operation is detected referring to the table, and r-bit detection result and x^r bit shift are EXORed r times to obtain the remainder R(x), so that branch processing is few in number in the CRC processing and a break is hardly occurred in the pipeline of DSP, enabling CRC operation to be efficiently executed. Furthermore, the CRC operation of parallel data allows higher speed processing compared with the CRC operation of serial data, enabling a processing time to be significantly shortened. In the CRC processing, the greater the

[Figure 10]

Figure 10 is a drawing illustrating an example of EXOR processing in Figure 6.

[Description of Symbols]

21 S/P converter

22 Register

30 DSP

31 Memory

32 EXOR operation section

33 CRC processing section

41 DSP interface

42 External circuit

11-4 REGISTER CLEAR

11-5 REGISTER CLEAR

11-6 REGISTER CLEAR

11-7 REGISTER CLEAR

11-12 REGISTER CLEAR

13-1 LATCH ENABLE

13-6 LATCH ENABLE

13-7 LATCH ENABLE

13-8 LATCH ENABLE

13-9 LATCH ENABLE

13-10 LATCH ENABLE

13-11 LATCH ENABLE

13-12 LATCH ENABLE

Figure 5

#1 PROCESSING CIRCUIT

#2 * BIT POSITION WHERE "1" IS SET UP IN GENERATOR POLYNOMIAL

#3 TERMINATION OF PROCESSING

#4 PROCESS FLOW OF CRC-32 PROCESSING

S1 INPUT 32-BIT DATA?

S2 STORE "crcdat" INTO MEMORY

S3 MASK PATTERN

S5 EXOR BETWEEN LSB BIT AND VALUE OF INPUT REGISTER

S6 SHIFT BY 32-BIT TOTAL (result 1) BY ONE BIT TO RIGHT

S7 FEED BACK LSB ONE BIT TO INPUT REGISTER

Figure 6

#1 START OF PROCESSING

#5 SHIFT BY FOUR BITS TO RIGHT

#6 EXOR

#7 OPERATION RESULT

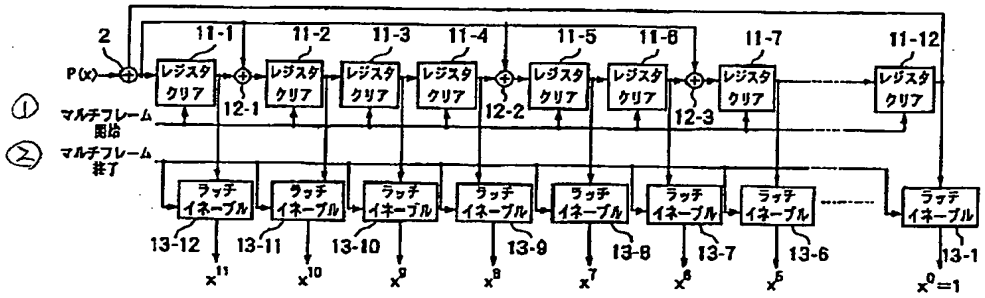
#8 SHIFT BY TWO BITS TO RIGHT

#9 SHIFT BY ONE BIT TO RIGHT

#10 ANSWER

#11 EXAMPLE OF EXOR PROCESSING IN FIGURE 6

【図４】 FIG.4



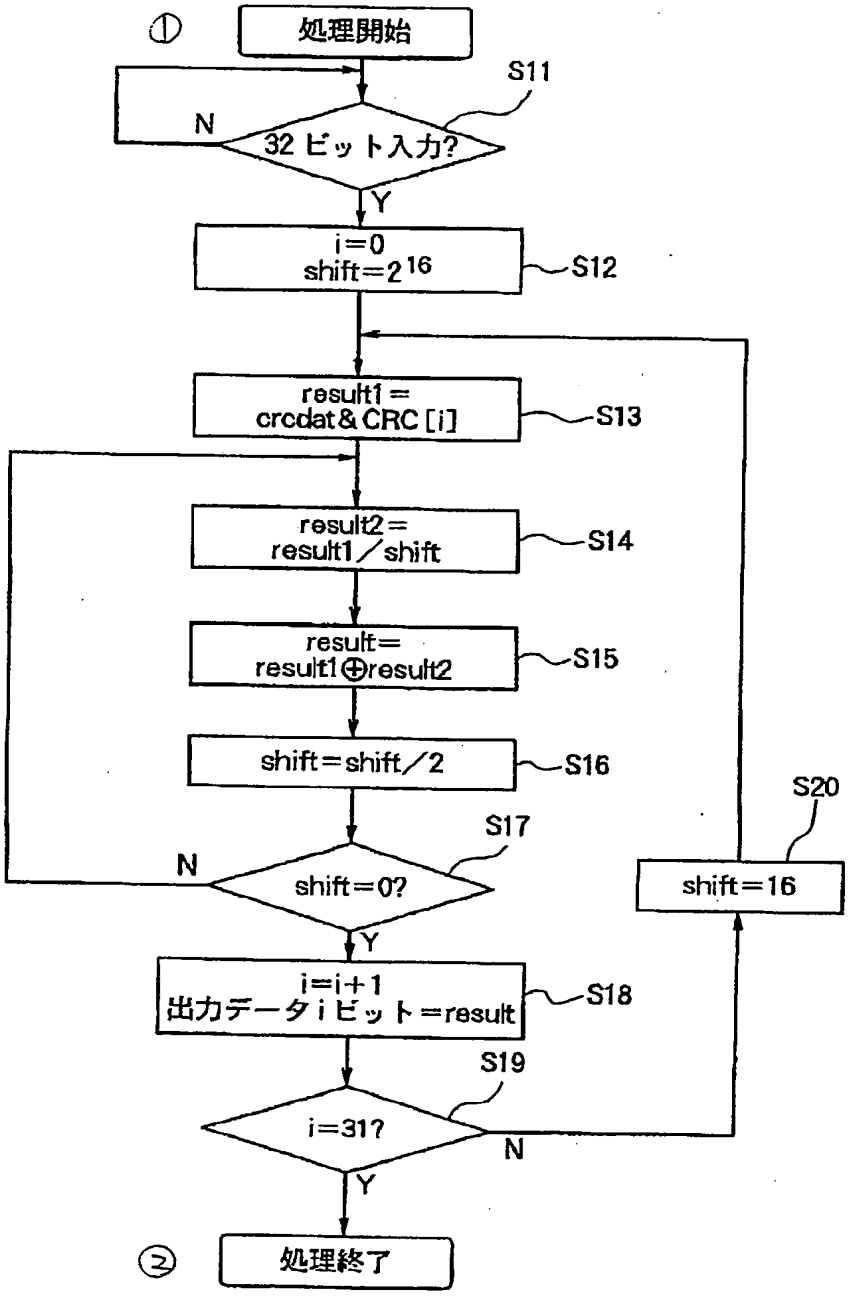
③ 図３中のCRC-12演算回路

【図７】 FIG.7

CRC	:	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	
AMARI_31	:	+31+ +31+ + +31+31+ +31+31+31+ + +31+31+ + + +31+31+31+31+ +31+ + + +31+31+31+ +31+	: ABE83D1D
AMARI_30	:	+30+30+ +30+ + + +30+ + + +30+ + +30+30+30+ + +30+ + + + + +30+ +30+ +30+ +30+	: D11330B5
AMARI_29	:	+29+29+29+ +29+ +29+ +29+29+29+ +29+ + +29+ +29+ + + +29+29+ +29+29+29+29+ + + +29+	: EAE946F1
AMARI_28	:	+28+28+28+28+ +28+28+28+ + + +28+ +28+ + + + +28+ +28+29+ +28+28+ +28+ + + +28+29+	: F7142DA9
AMARI_27	:	+27+27+27+27+27+ + +27+27+27+27+ +27+ +27+ +27+ + +27+27+ + + + + + + +27+ +27+ +	: F9EA98BA
AMARI_26	:	+ +26+26+26+26+26+ + +26+26+26+26+ +26+ +29+ +26+ + +26+26+ + + + + + + +26+ +26+	: 7CF54C86
AMARI_25	:	+25+ +25+25+25+25+ + + + + +25+25+ +25+ + + +25+ +25+ + + +25+25+ +25+25+ + +25+	: 8C1A2BD9
AMARI_24	:	+24+24+ +24+24+24+ + + +24+24+ +24+24+ +24+24+ + +24+24+ +24+ +24+ +24+24+ +24+24+24+	: DC809957
AMARI_23	:	+23+23+23+ +23+23+ + + +23+ +23+ +23+23+ + +23+ + + + +23+23+23+ + + + + + + +	: EC584386
AMARI_22	:	+ +22+22+22+ +22+22+ + + +22+ +22+ +22+22+ + +22+ + + + +22+22+22+ + + + + + +	: 7B2B21D9
AMARI_21	:	+ + +21+21+21+ +21+21+ + + +21+ +21+ +21+21+ + +21+ + + + +21+21+21+ + + + + +	: 88153BEB
AMARI_20	:	+ + + +20+20+20+ +20+20+ + + +20+ +20+ +20+20+ + +20+ + + + +20+20+20+ + + + +	: 106AC87F
AMARI_19	:	+ + + + +19+19+19+ +19+19+ + + +19+ +19+ +19+19+ + +19+ + + + + +19+19+19+ + + +	: 8CD56488
AMARI_18	:	+ + + + + +19+19+19+ +19+19+ + + +19+ +19+ +19+19+ + +19+ + + + +19+19+19+ + + +	: 87623210
AMARI_17	:	+ + + + + + +17+17+17+ +17+17+ + + +17+ +17+ +17+17+ + +17+ + + + +17+17+17+ + + +	: 83B1589E
AMARI_16	:	+ + + + + + + +16+16+16+ +16+16+ + + +16+ +16+ +16+16+ + +16+ + + + +16+16+16+	: 810BAC87

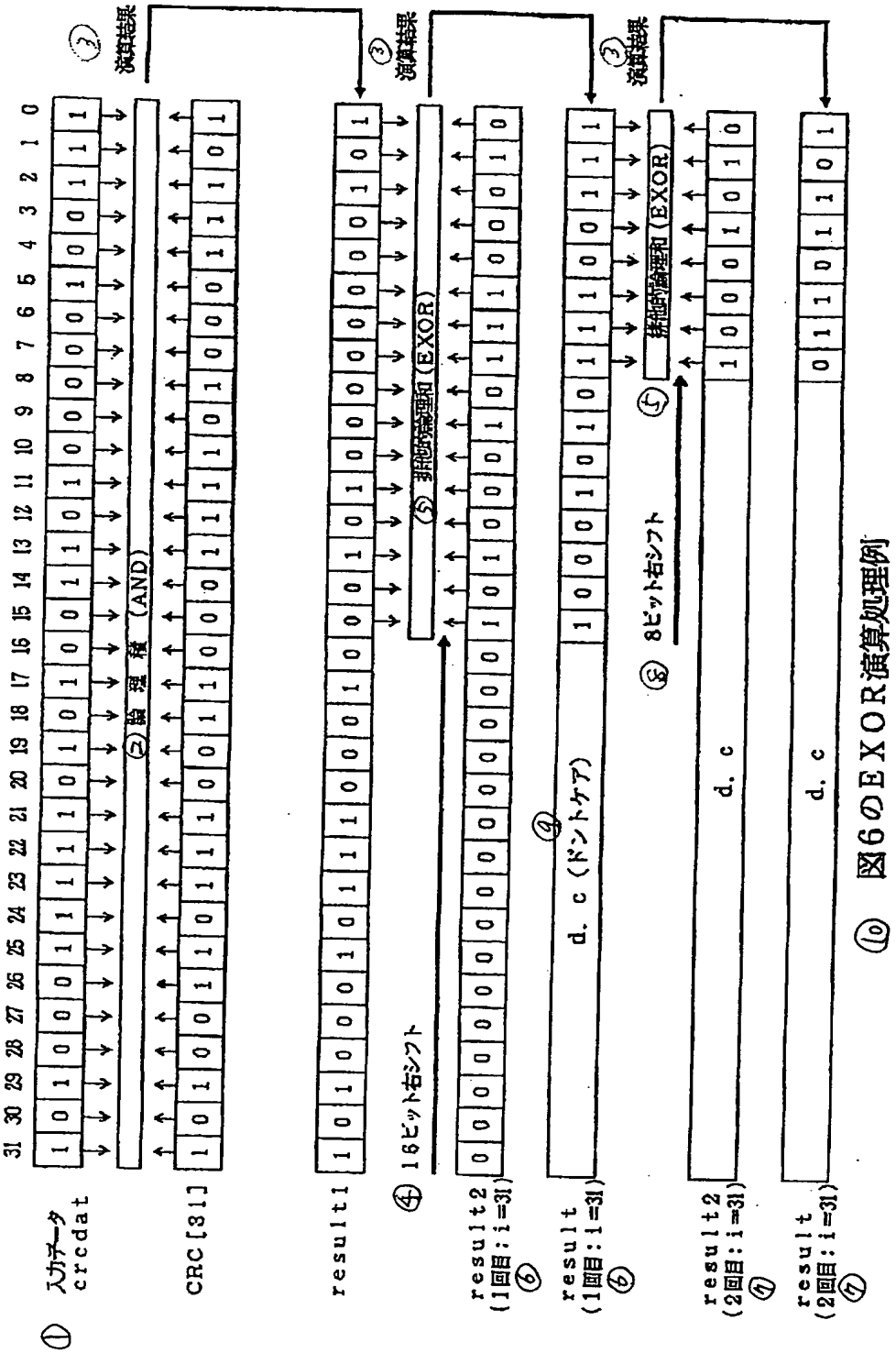
① 図１のＣＲＣ－３２演算用テーブル

【図6】 FIG. 6



③ 図1のCRC-32演算の処理フロー

【図9】 FIG. 9


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
① 入力データ crcdat
② 論理積 (AND)
CRC[31]
演算結果
③ 演算結果
result1
④ 16ビット右シフト
⑤ 排他的論理和 (EXOR)
result2 (1回目：i=31)
⑥
result (1回目：i=31)
d. c (ドントケア)
⑧ 8ビット右シフト
⑤ 排他的論理和 (EXOR)
result2 (2回目：i=31)
⑦
result (2回目：i=31)
d. c
⑩ 図6のEXOR演算処理例